Heaps Exercises

Yousef Alaa Awad

1

In an array-based implementation of a Heap, the left-child of the left-child of the node at index i, if it exists, can be found at what array location?

I will assume we start at the node, therefore it will be the following:

$$2 * (2i + 1) + 1 \rightarrow 4i + 2 + 1 \rightarrow 4i + 3$$

2

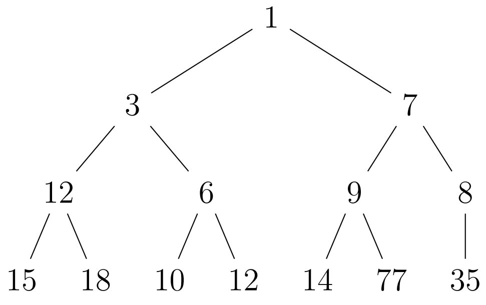
In an array-based implementation of a Heap, the right-child of the right-child of the node at index i, if it exists, can be found at what array location?

I will assume we start at the node, therefore it will be the following:

$$2 * (2i + 2) + 2 \rightarrow 4i + 4 + 2 \rightarrow 4i + 6$$

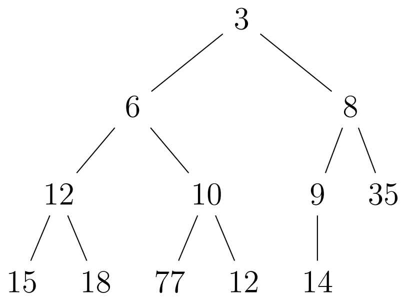
3

Show the result of inserting the item 7 into the heap shown below:



4

Show the result of removing the minimum element from the original heap in question #2 (without 7) from above.



5

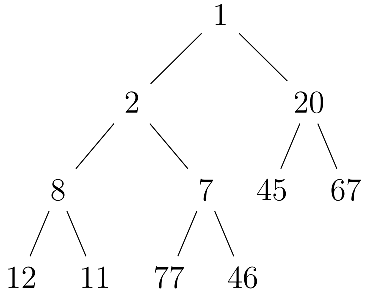
Show the array representation of the original heap from question #2.

1	3	8	12	6	9	35	15	18	10	12	14	77

6

Run the whole Heapify function on the following random values: (this is the function that builds a heap in O(n) time)

I will assume we want to turn it into a min heap...



7

Explain each step shown in the code below, for the percolateDown function:

```c
void percolateDown(struct heapStruct *h, int index) {
  int min; // minimum index we found.
  if ((2*index+1) <= h->size) { // if the index given has 2 children, do the following
      min = minimum(h->heaparray[2*index], 2*index, h->heaparray[2*index+1], 2*index+1); // calling a function to find the minimum child's index.
    if (h->heaparray[index] > h->heaparray[min]) { // minimum child found above is smaller than the current one we are on.
      swap(h, index, min); // if it is bigger, swap them.
      percolateDown(h, min); // call again with the minimum index, to make sure it does not need to be percolated down.
    }
  } else if (h->size == 2*index) { // if the index given only has 1 child.
   if (h->heaparray[index] > h->heaparray[2*index]) // check if the current one is larger than its child
    swap(h, index, 2*index); // and if so, swap them.
   }
  }
```